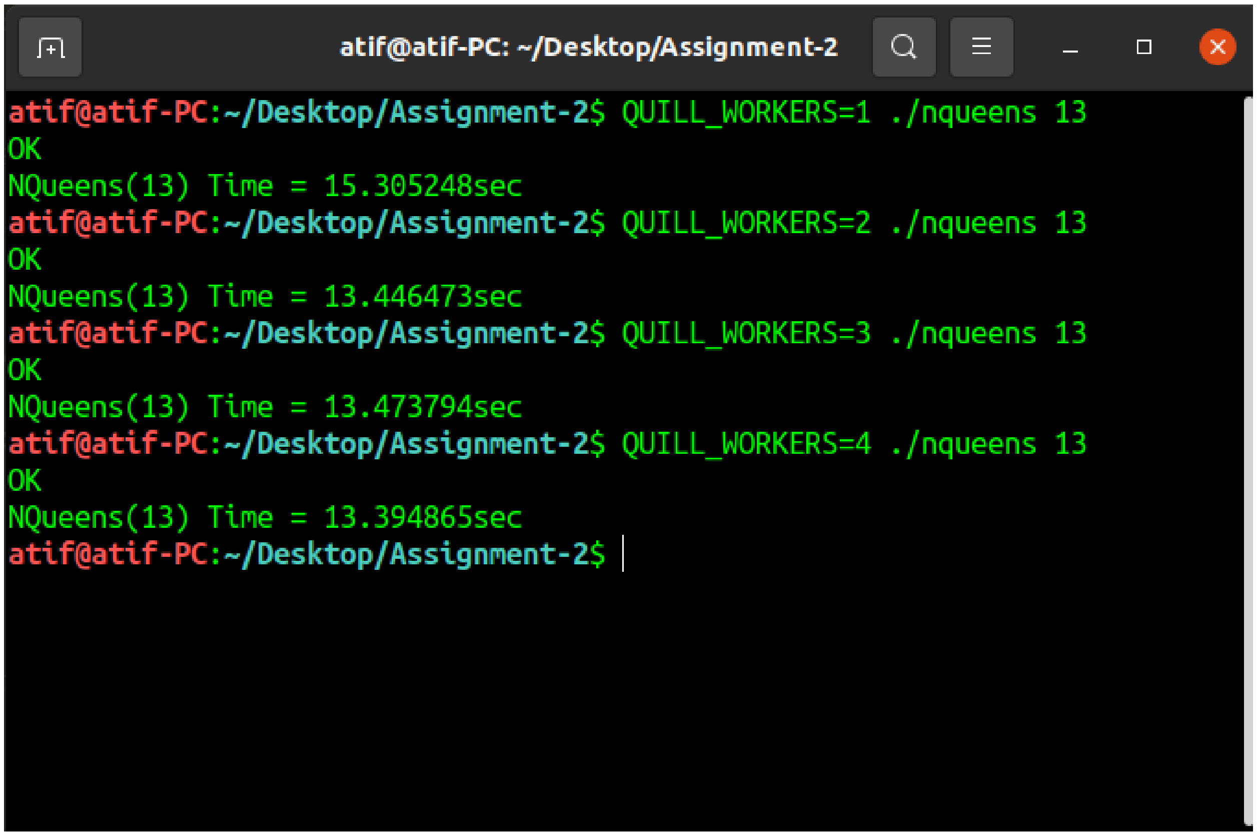
QUILL: A light-weight work-stealing runtime for async-finish parallelism

Observations:-

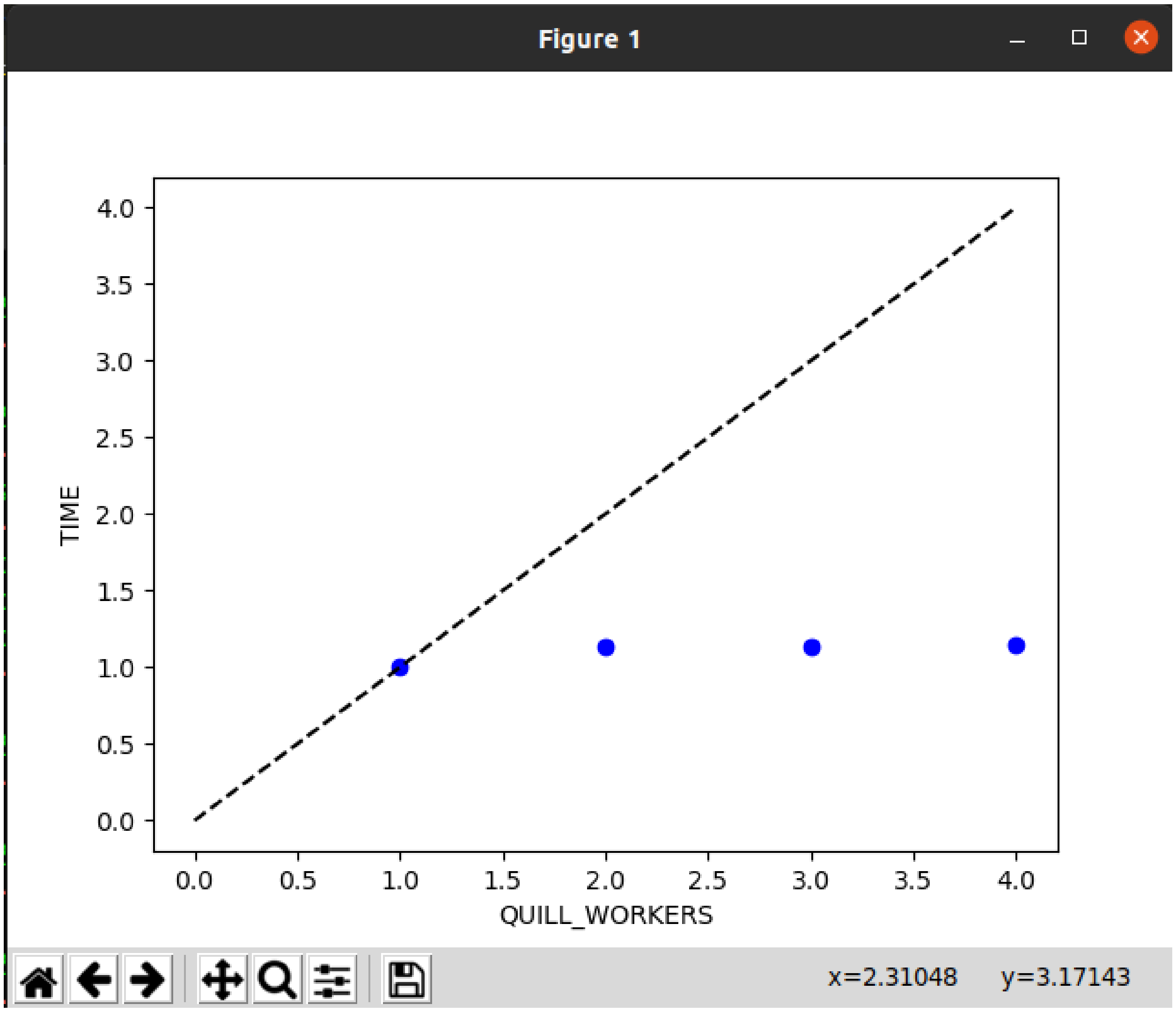
The benchmark with the workers is 1, 2, 3, 4, and parameters for nqueens being 13 are as follows:-



```
atif@atif-PC:~/Desktop/Assignment-2$ QUILL_WORKERS=1 ./nqueens 13
OK
NQueens(13) Time = 15.305248sec
atif@atif-PC:~/Desktop/Assignment-2$ QUILL_WORKERS=2 ./nqueens 13
OK
NQueens(13) Time = 13.446473sec
atif@atif-PC:~/Desktop/Assignment-2$ QUILL_WORKERS=3 ./nqueens 13
OK
NQueens(13) Time = 13.473794sec
atif@atif-PC:~/Desktop/Assignment-2$ QUILL_WORKERS=4 ./nqueens 13
OK
NQueens(13) Time = 13.394865sec
atif@atif-PC:~/Desktop/Assignment-2$ 
```

Time required to execute the program with QUILL_WORKERS = 1: 15.305248sec
Time required to execute the program with QUILL_WORKERS = 2: 13.446473sec
Time required to execute the program with QUILL_WORKERS = 3: 13.473794sec
Time required to execute the program with QUILL_WORKERS = 4: 13.394865sec

The speedup graph is plotted using the matplotlib library in python and is shown below.
The speedup is calculated using the formula :
T(linear)/T(parallel)



We can see a clear speedup when QUILL_WORKERS are changed from 1 to 2, as now the task is being distributed between two threads. It is not that the task is equally distributed among the workers. The distribution is uneven. Then we see then that there is a speedup from QUILL_WORKERS = 3 to QUILL_WORKERS = 4.

The speedup is sub-linear and not super-linear as the points plotted are below the line y = x.

Also, the speedup is not shown much due to the unavailability of resources as the program was run on a VM, and the cores on the VM are insufficient to show the speedup for QUILL_WORKERS > 2.